Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота №3
Технології розробки програмного забезпечення
«ДІАГРАМА РОЗГОРТАННЯ. ДІАГРАМА КОМПОНЕНТІВ.
ДІАГРАМА ВЗАЄМОДІЙ ТА ПОСЛІДОВНОСТЕЙ»

Виконала:
студент групи ІА-24
Красношапка Р. О.
Перевірив:
Мягкий М. Ю.

Київ 2024

Зміст

Тема: ДІАГРАМА РОЗГОРТАННЯ. ДІАГРАМА КОМПОНЕНТІВ. ДІАГРАМА ВЗАЄМОДІЙ ТА ПОСЛІДОВНОСТЕЙ.

Мета: Розробити діаграми розгортання, компонентів, взаємодій та послідовностей для обраної тема проекту

Короткі теоретичні відомості

Діаграма розгортання (Deployment Diagram)

Мета:

Діаграма розгортання в UML використовується для моделювання фізичного розташування програмного забезпечення і апаратного забезпечення. Вона показує, як компоненти системи розгортаються на вузлах (сервери, клієнти тощо).

Ключові елементи:

- **Вузли (Nodes):** Фізичні пристрої або віртуальні сервери, де розгортається програмне забезпечення.
- **Артефакти (Artifacts):** Програмні модулі (наприклад, файли, бази даних), які розгортаються на вузлах.
- **Зв'язки (Communication Paths):** Лінії між вузлами, що показують обмін даними.

Приклад застосування:

- Розподіл серверних компонентів між веб-сервером, базою даних і клієнтськими пристроями.

Діаграма компонентів (Component Diagram)

Мета:

Діаграма компонентів використовується для представлення структури системи з точки зору її модулів або компонентів. Вона показує, як ці компоненти взаємодіють один з одним і з зовнішнім середовищем.

Ключові елементи:

- **Компоненти (Components):** Логічні модулі, що виконують певні функції.
- **Інтерфейси (Interfaces):** Визначають точки взаємодії компонентів.
- **Залежності (Dependencies):** Показують, як один компонент залежить від іншого.

Приклад застосування:

- Визначення зв'язків між мікросервісами у розподіленій архітектурі.

Діаграма взаємодій (Interaction Diagram)

Мета:

Діаграма взаємодій відображає, як об'єкти системи взаємодіють між собою для виконання певного сценарію.

Типи:

1. **Діаграма послідовностей (Sequence Diagram):**
    - **Мета:** Показує порядок взаємодії об'єктів у часі.
    - **Ключові елементи:**
        - **Актори (Actors):** Ініціюють взаємодії.
        - **Об'єкти (Objects):** Елементи системи, які беруть участь у процесі.
        - **Повідомлення (Messages):** Вказують передачу даних між об'єктами.
    - **Приклад:** Авторизація користувача в системі.
2. **Діаграма комунікацій (Communication Diagram):**
    - **Мета:** Фокусується на зв'язках між об'єктами.
    - **Ключові елементи:**
        - **Вузли (Nodes):** Об'єкти, які взаємодіють.
        - **Повідомлення:** Лінії, які показують передачу даних між вузлами.

Приклад застосування:

- **Діаграма послідовностей:** Моделювання процесу оформлення замовлення.
- **Діаграма комунікацій:** Представлення зв'язків між сервісами в мікросервісній архітектурі.

Ці діаграми в поєднанні допомагають зрозуміти архітектуру системи, її компоненти, фізичну інфраструктуру та взаємодію між різними елементами в часі.

Хід роботи

Project Management software (proxy, chain of responsibility, abstract factory, bridge, flyweight, client-server) Програмне забезпечення для управління проектами повинно мати наступні функції: супровід завдань/вимог/проектів, списків команд, поточних завдань, планування за методологіями agile/kanban/rup (включаючи дошку завдань, ітерації тощо), мати можливість прикріплювати вкладені файли до завдань та посилатися на конкретні версії програми, зберігати виконувані файли для кожної версії.

Діаграма розгортання для проектованої системи

На діаграмі розгортання зображаємо фізичні компоненти, необхідні для роботи системи. Це сервер, на якому буде встановлено Backend та базу данних

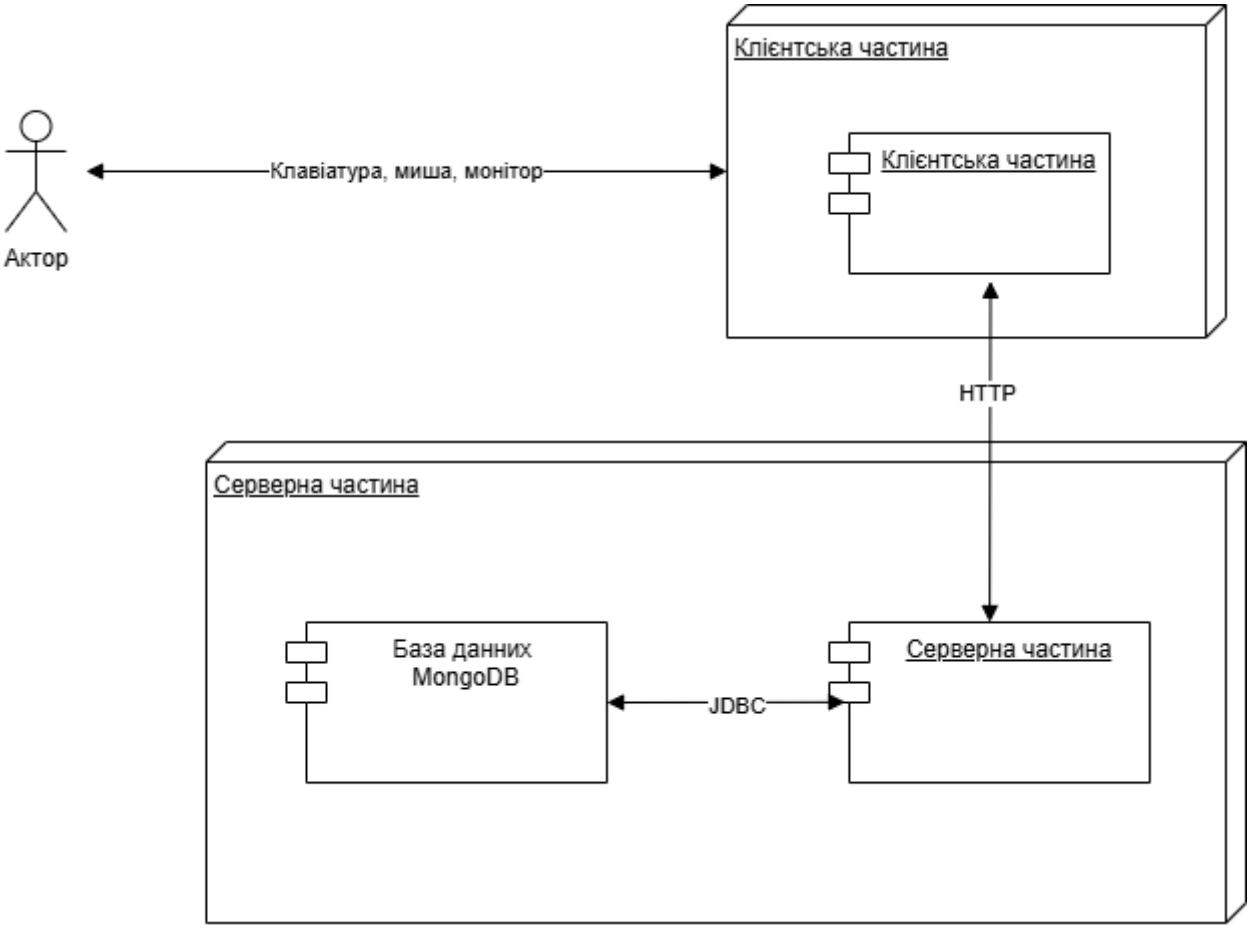

Рисунок 1 — Діаграма розгортання

Опис компонентів:

1. **Actor (Користувач):**
    - Представляє зовнішнього користувача системи.
    - Використовує клієнтську частину системи через клавіатуру, мишу і монітор.

2. **Клієнтська частина:**
    - Відповідає за взаємодію з користувачем.
    - Надсилає запити до серверної частини через протокол HTTP.
    - Містить інтерфейс для введення і відображення даних.
3. **Серверна частина:**
    - Основний компонент для обробки логіки додатка.
    - Приймає HTTP-запити від клієнтської частини, обробляє їх і надсилає відповіді.
    - Використовує JDBC (Java Database Connectivity) для взаємодії з базою даних.
4. **База даних (MongoDB):**
    - Зберігає дані системи.
    - Взаємодіє з серверною частиною через JDBC.

Взаємодії:

- **Actor -> Клієнтська частина:** Користувач взаємодіє з клієнтською частиною через пристрої введення (клавіатура, миша) та пристрої відображення (монітор).
- **Клієнтська частина -> Серверна частина:** Клієнтська частина надсилає HTTP-запити до серверної частини для виконання бізнес-логіки та отримання даних.
- **Серверна частина -> База даних:** Серверна частина здійснює запити до бази даних через JDBC для зчитування, оновлення або збереження даних.

Основні протоколи та інструменти:

- **HTTP:** Використовується для передачі даних між клієнтською і серверною частиною.
- **JDBC:** Протокол для підключення серверної частини до бази даних MongoDB.

Ця архітектура ілюструє типову структуру клієнт-серверного додатка із базою даних. Якщо потрібна додаткова інформація або уточнення, повідомте!

Діаграма компонентів для розроблюваної системи

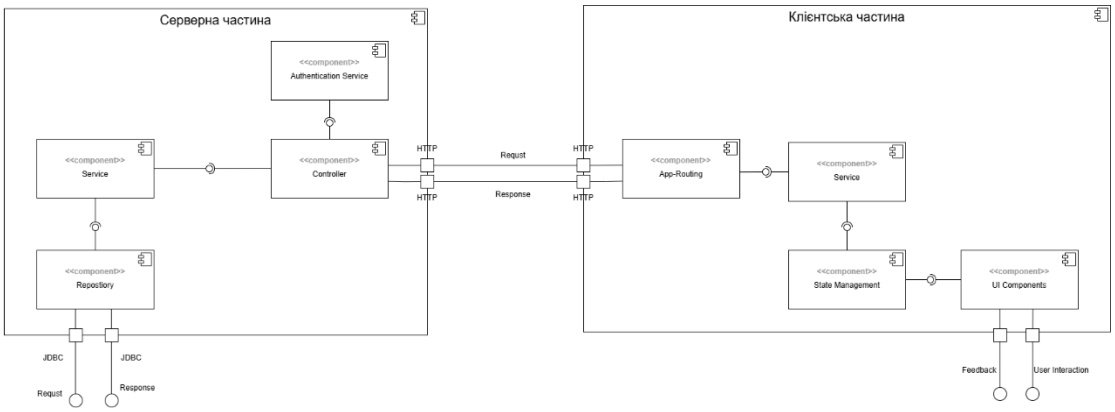

Рисунок 2 — Діаграма компонентів

UML-діаграма компонентів, яка ілюструє архітектуру системи. Вона поділена на дві основні частини:

1. **Серверна частина ("Серверна частина")**:
    - **Authentication Service** (Сервіс автентифікації): Виконує логіку автентифікації.
    - **Controller** (Контролер): Слугує проміжним шаром, який обробляє HTTP-запити та передає їх відповідному сервісу.
    - **Service** (Сервіс): Представляє бізнес-логіку застосунку.
    - **Repository** (Репозиторій): Відповідає за взаємодію з базою даних за допомогою JDBC, виконуючи запити та обробляючи відповіді.
2. **Клієнтська частина ("Клієнтська частина")**:
    - **App-Routing** (Маршрутизація додатку): Управляє навігацією та маршрутизацією між компонентами застосунку.
    - **Service** (Сервіс): Здійснює логіку, пов’язану із клієнтською частиною застосунку, можливо, взаємодіючи із сервером.
    - **State Management** (Управління станом): Забезпечує збереження стану застосунку та узгодженість даних між компонентами інтерфейсу.
    - **UI Components** (Компоненти інтерфейсу): Представляють елементи інтерфейсу, які забезпечують взаємодію з користувачем та відображають зворотний зв'язок.

Компоненти пов’язані через HTTP-запити та відповіді, а взаємодія користувача відображена у вигляді циклів вводу та зворотного зв’язку.

Діаграма послідовностей для проектованої системи

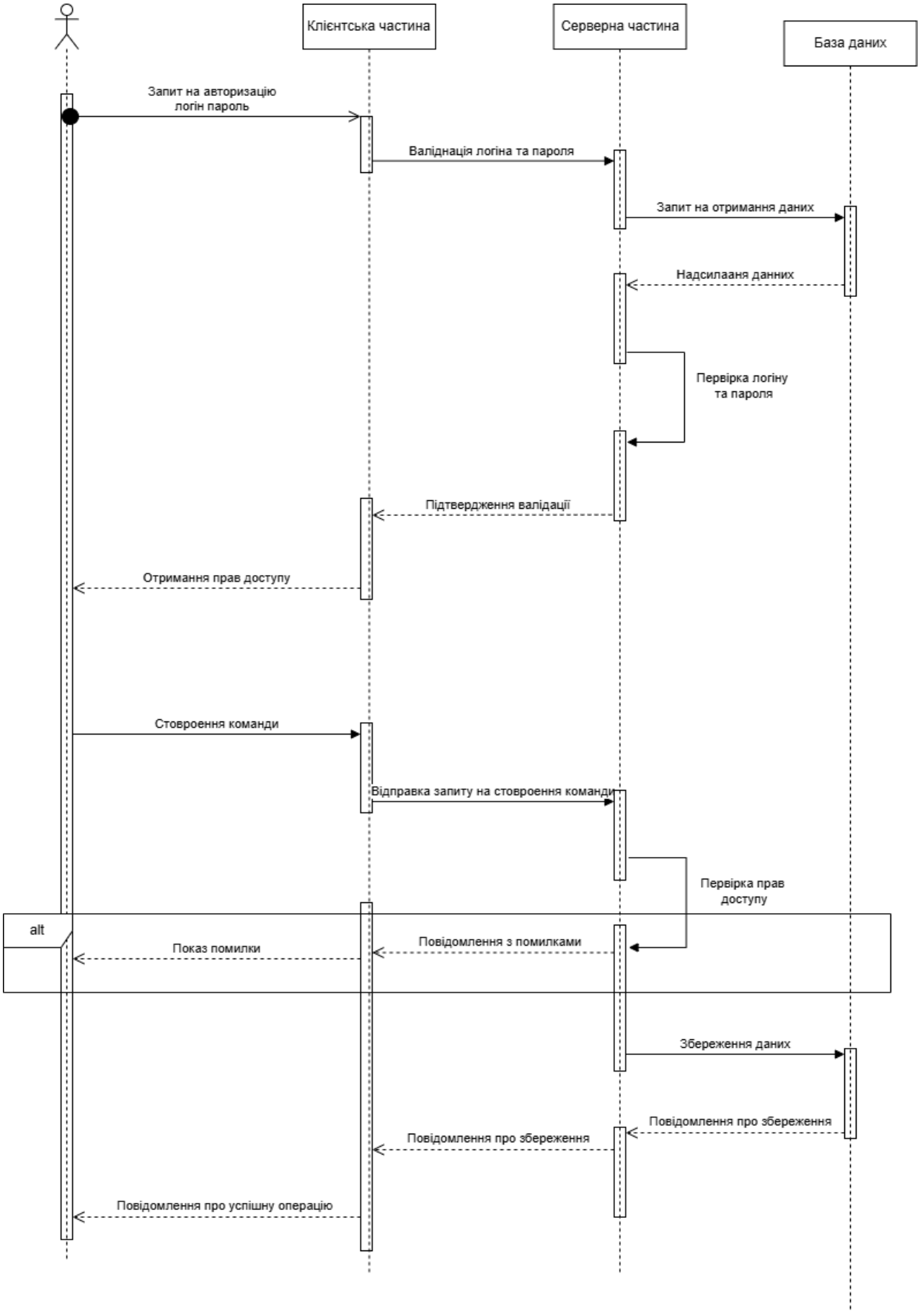

Рисунок 3 — Діаграма послідовностей

Діаграма послідовностей (sequence diagram), яка демонструє процес взаємодії між клієнтською частиною, серверною частиною та базою даних у контексті авторизації та створення команди.

Опис процесу:

1. **Авторизація:**
    - Користувач вводить логін і пароль, які надсилаються на клієнтську частину.
    - Клієнтська частина надсилає запит на сервер для валідації логіна і пароля.
    - Сервер виконує запит до бази даних для перевірки введених даних.
    - Якщо перевірка успішна, сервер надсилає підтвердження валідації клієнту.
    - Користувач отримує права доступу.
2. **Створення команди:**
    - Користувач ініціює процес створення команди.
    - Клієнтська частина відправляє запит на створення команди до серверної частини.
    - Сервер перевіряє права доступу користувача.
    - Якщо права коректні:
        - Сервер зберігає дані в базі.
        - База даних надсилає підтвердження збереження.
        - Користувач отримує повідомлення про успішну операцію.
    - Якщо виникає помилка:
        - Сервер надсилає повідомлення про помилку.
        - Користувач бачить повідомлення про помилку.

Умови:

- **alt**: Обробляє випадки, коли операція може завершитися помилкою або успіхом.

Ця діаграма ілюструє чіткий потік операцій і забезпечує розуміння ключових кроків у роботі системи.

Посилання на репозиторій:

https://github.com/QUIRINO228/projectManagmentSoftware

Висновок: У даній лабораторній роботі було створено діаграми послідовностей, розгортання та компонентів. Це дозволяє краще зрозуміти архітектуру системи, процеси її роботи та взаємодію між складовими.

Діаграма послідовностей деталізує взаємодії між користувачем, клієнтською частиною, сервером і базою даних, визначаючи основні етапи авторизації, перевірки прав доступу та створення даних у системі. Особливу увагу приділено сценаріям обробки помилок.

Діаграма розгортання відображає фізичне розташування компонентів системи, взаємодію між клієнтською та серверною частинами, а також підключення до бази даних. Це дає змогу зрозуміти, як система функціонує у реальному середовищі.

Діаграма компонентів описує логічну структуру системи, включаючи модулі, сервіси, контролери та взаємодію з базою даних. Вона допомагає зрозуміти, як програмні частини об'єднуються в єдину архітектуру, і показує функції кожного модуля.

Створені діаграми є основою для документування архітектури системи, полегшують комунікацію між розробниками та зацікавленими сторонами, а також сприяють побудові технічного завдання та оптимізації процесів